UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR



Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

# TRABAJO FIN DE GRADO

## DETECCIÓN DE OBJETOS ABANDONADOS PARA VÍDEO-VIGILANCIA A LARGO PLAZO

**Sergio López Álvarez**

**Tutor: Diego Ortego Hernández**

**Ponente: José María Martínez Sánchez**

**Mayo 2016**

# Detección de objetos abandonados para vídeo-vigilancia a largo plazo

**Sergio López Álvarez**
**Tutor: Diego Ortego Hernández**
**Ponente: José María Martínez Sánchez**

**Video Processing and Understanding Lab**
**Departamento de Tecnología Electrónica y de las Comunicaciones**
**Escuela Politécnica Superior**
**Universidad Autónoma de Madrid**
**Mayo 2016**

**Trabajo parcialmente financiado por el gobierno español bajo el proyecto HAVideo TEC2014-53176-R**

# Resumen

En este trabajo se ha integrado un algoritmo de detección de objetos abandonados en una plataforma de procesado distribuido de vídeo denominada DiVA. Para poder realizar esta integración y además mejorar la velocidad de procesado del sistema, se ha estudiado en profundidad el funcionamiento de dicho algoritmo y las distintas plataformas y herramientas utilizadas, para posteriormente implementar su funcionalidad en C++ y operar aproximadamente en tiempo real. Además, se propone un nuevo algoritmo de detección de objetos abandonados que mejora las capacidades del sistema base. Estas mejoras consisten en la capacidad de filtrar falsas detecciones valiéndose de la teoría de entropía y de la eliminación de detecciones originadas por personas estáticas. Finalmente, se ha evaluado el rendimiento del sistema implementado con respecto al sistema base de partida, tanto en lo relativo a coste computacional, como en la precisión de las detecciones de objetos abandonados, consiguiendo mejorar en ambos aspectos.

### Palabras clave

Cuadro, agrupamiento temporal de bloques, blob, objetos estáticos, cluster, filtrado, detecciones,objetos abandonados, optimización, trabajo en tiempo real.

# Abstract

The main objective of this project has been the integration of an abandoned object detection algorithm on a distributed video analysis platform called DiVA. To achieve this, and also improve the processing speed of the system, the base algorithm's operation and all the platforms and tools used in this project have been deeply studied, in order to implement its functionality on C++ later, adapting it so that the system could operate not only using *off-line* videos, but also in real time. Moreover, this project also proposes a new abandoned objects detection algorithm that improves the capabilities of the base system. This improvements allow the new system to filter false detections using entropy theory and also avoid detections due to static humans. Finally, the performance of the system has been thoroughly evaluated, both according to computational cost and the precision of the detections, proving a noticeable improvement in both aspects.

**keywords**

Frame, online block clustering, blob, static objects, cluster, filtering, detections, blocks, abandoned objects, optimization, real time operation.

# Agradecimientos

*Quiero agradecer en primer lugar a Diego, mi tutor, por haberme elegido para llevar a cabo este proyecto, haberme ayudado a superar todos los problemas que han surgido a lo largo del desarrollo del mismo y haber tenido siempre la paciencia suficiente para explicarme todas las cosas que han sido nuevas para mi a la hora de realizar un proyecto de estas características por primera vez.*

*También quiero agradecer a todos los amigos que me dejan estos años de carrera, especialmente a Edu, Tony, Carol, Pilar y a otros muchos, compañeros de prácticas y de estudio sin los cuales probablemente no estaría escribiendo esto, así como a mi compañero de piso y amigo Víctor, que me aguanta en casa desde hace ya cuatro años.*

*Además, me gustaría hacer una mención especial a toda la comunidad on-line del foro Stackoverflow, fuente infinita de ayuda en lo que se refiere a problemas de programación, y muy especialmente al usuario "Dipak Ingole" por su utilísimo post sobre matrices tridimensionales.*

*Y por último, agradecer a toda mi familia, que me ha apoyado siempre en mi afán de estudiar esta carrera, a mis padres Anabel y Fernando y a mis hermanos Jaime y Guillermo. Agradecer especialmente a mi madre, siempre pendiente de mí en todos los sentidos, y sin cuyo apoyo y consejo no podría haber sobrevivido a todas esas épocas de exámenes.*

Gracias a todos.
Sergio López Álvarez
Mayo 2016

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

En este capítulo vamos a introducir la motivación del desarrollo de este trabajo y la situación actual del campo de detección de objetos abandonados, dentro de la investigación en tratamiento de vídeo, así como los objetivos del proyecto y la estructura general que sigue este documento.

## 1.1. Motivación

Durante los últimos años, debido al abaratamiento de los sistemas de videovigilancia distribuidos y su avance tecnológico, muchos algoritmos de tratamiento de imagen han comenzado a salir de los entornos de desarrollo controlados, extendiéndose su aplicación a entornos reales. Sin embargo, la configuración de la mayoría de sistemas es muy dependiente del entorno de aplicación. Esta dependencia afecta negativamente a su rendimiento a medio y largo plazo en los distintos entornos de aplicación (estaciones de tren, centros comerciales u otros lugares públicos) donde la adaptación tanto a variaciones bruscas como a graduales de la escena es un requisito crucial para conseguir operar de forma fiable.

La detección de objetos abandonados en entornos públicos y privados ha centrado un gran interés en el ámbito de la visión artificial durante los últimos años [8]. Esta tarea consiste en detectar todo objeto que, tras haber estado en movimiento, permanece parado o estático en la escena. No obstante, operar a largo plazo de forma fiable es aún un reto a resolver que limita la utilización de estos algoritmos en entornos reales. Típicamente, los trabajos desarrollados en el estado del arte llevan a cabo una detección de los objetos de la escena mediante una técnica de segmentación frente-fondo o *Background Subtraction* (BS) [4], para después determinar qué objetos están abandonados. Estas técnicas están condicionadas por las limitaciones del algoritmo

de BS para adaptarse a las variaciones de la escena.

Por tanto, la motivación principal de este trabajo es contribuir a la detección de objetos abandonados, donde estudios recientes muestran la dificultad de operar en entornos reales. Detectar situaciones de abandono de objetos es de gran interés para prevenir situaciones potencialmente peligrosas (por ejemplo atentados terroristas), al ser una herramienta capaz de centrar la atención del personal de video-seguridad en posibles zonas de interés.

## 1.2. Objetivos

El objetivo de este TFG es la integración y mejora de un algoritmo de detección de objetos abandonados en un sistema de análisis distribuido de video. El objetivo inicial se divide en los siguientes sub-objetivos:

1. Estudio del sistema base: En esta etapa se analizará el sistema base desarrollado en MATLAB [12], para comprender su funcionamiento a nivel conceptual y de implementación. El objetivo es conocer tanto las debilidades del algoritmo, como su funcionamiento detallado, para poder abordar sus limitaciones e implementar el algoritmo en un nuevo lenguaje.

2. Adaptación del algoritmo base: En esta etapa se adaptará el algoritmo base implementado en la plataforma MATLAB a C++ empleando la librería de tratamiento de imagen y vídeo OpenCV. El objetivo es desarrollar un código que permita operar a una velocidad cercana al tiempo real.

3. Mejoras del sistema base: Esta etapa consiste en modificar el sistema base para abordar sus limitaciones. El objetivo es mejorar el comportamiento a largo plazo del algoritmo mediante la eliminación de falsas detecciones.

4. Encapsulación del algoritmo en una plataforma de análisis distribuido: En esta etapa se realizará la encapsulación del algoritmo en C++ dentro de la plataforma DiVA (*Distributed Video Analysis*), siguiendo los distintos niveles de integración especificados por la misma, para poder utilizar el algoritmo en un sistema de cámaras distribuidas.

5. Análisis de resultados: En esta etapa se van a analizar y comparar los resultados obtenidos por el sistema base y el algoritmo desarrollado, tanto para las detecciones realizadas, como para el coste computacional.

## 1.3. Estructura de la memoria

La memoria del proyecto se divide en los siguientes capítulos:

- Capítulo 1: Introducción, motivación del trabajo y objetivos.
- Capítulo 2: Estado del arte sobre sistemas de detección de objetos abandonados y la plataforma DiVA.
- Capítulo 3: Diseño y desarrollo de la implementación e integración del sistema base en C++ y DiVA.
- Capítulo 4: Mejoras del sistema base.
- Capítulo 5: Resultados experimentales.
- Capítulo 6: Conclusiones y trabajo futuro.

# Capítulo 2

# Estado del arte

Este capítulo se encuentra dividido en tres secciones. En primer lugar, se proporciona una visión general del trabajo previo en el área de la detección de objetos abandonados, a través de otros sistemas importantes pertenecientes a la literatura. A continuación, se presenta un estudio en detalle del sistema base de partida de este trabajo (Sección 2.2), y finalmente se introduce el sistema de análisis distribuido de vídeo DiVA (Sección 2.3).

## 2.1. Sistemas previos en la literatura

La detección de objetos abandonados ha experimentado recientemente un elevado interés investigador [8], debido a su utilidad a la hora de prevenir ataques terroristas detectando objetos abandonados [11] o vehículos aparcados ilegalmente [1]. Los algoritmos de detección de objetos abandonados (AOD), están basados en los algoritmos de detección de objetos estáticos (SOD), que detectan la estaticidad en una escena después de cierto movimiento previo.

Típicamente, un algoritmo de sustracción de fondo o *Background Substraction* (BS), se encarga de separar los objetos (frente) del fondo, y el algoritmo SOD decide si estos son o no objetos estáticos [2]. Sin embargo, los algoritmos recientes de BS tienen numerosos errores al operar en entornos reales [4], penalizando el rendimiento de los algoritmos SOD.

Por un lado, los algoritmos SOD más recientes suelen utilizar estrategias basadas en BS. Por ejemplo, están muy extendidas las aproximaciones basadas en la acumulación temporal del frente de la escena [9], donde el postprocesado [10][14] y la combinación de características adicionales [13] son necesarias para lidiar con las dificultades de los algoritmos de BS en presencia de multitudes o cambios de iluminación.

Además existen aproximaciones que, junto con información de movimiento, realizan un muestreo temporal del frente de la escena para poder operar en entornos complejos [3], siendo la frecuencia de muestreo crítica para mantener un rendimiento adecuado. También existen las estrategias basadas en dos modelos de BS (Dual BS), que dependen de la capacidad del algoritmo de BS para adaptarse rápidamente a los cambios de la escena y así poder detectar objetos estáticos [11][15]. Sin embargo, la sustracción de fondo provoca errores que requieren un postprocesado adicional, como el uso de características de bordes o la interacción de ambos modelos de BS [7], para evitar las falsas detecciones ocasionadas por objetos pertenecientes al fondo que son retirados de la escena. Otras aproximaciones utilizan algoritmos de BS multicapa para modelar conjuntamente objetos móviles, estáticos y del fondo [18]. A pesar de ello, sigue siendo necesario utilizar etapas de validación de los candidatos detectados mediante características de región [8] y bordes [16] para poder lidiar con los posibles errores cometidos por los algoritmos de BS.

Por otro lado, los algoritmos de AOD clasifican cada objeto estático como abandonado, robado o persona. En [14], las falsas detecciones se detectan empleando el gradiente de los contornos de los objetos en la imagen, que suele ser mayor para objetos abandonados que para robados. También existen aproximaciones como [17], que analizan el contexto que rodea a un objeto junto con un detector de personas para detectar objetos abandonados a partir de objetos estáticos. Otras aproximaciones relevantes como [11], abordan la tarea de AOD empleando *back-tracking* para detectar cuándo el dueño de un objeto se ha alejado de éste, mientras que en [8] se lleva a cabo un *ranking* de las alarmas en función de su relevancia para determinar aquellas con menor puntuación como falsas.

Muchos de los problemas de los algoritmos de AOD de la literatura, están relacionados con los problemas de los algoritmos de BS para manejar cambios de iluminación, multitudes o movimientos intermitentes de objetos, que requieren una capacidad de adaptación temporal. Estos aspectos son muy importantes en situaciones reales, donde puede ser necesario que el algoritmo trabaje de forma continuada, es decir que opere a largo plazo.

## 2.2. Sistema base

### 2.2.1. Estructura general

El algoritmo base detecta objetos estáticos sin necesidad de utilizar un algoritmo de sustracción de fondo (ver Figura 2.1), técnicas que tal y como se ha mencionado en la literatura, presentan numerosos problemas. En su lugar, se detectan cambios de

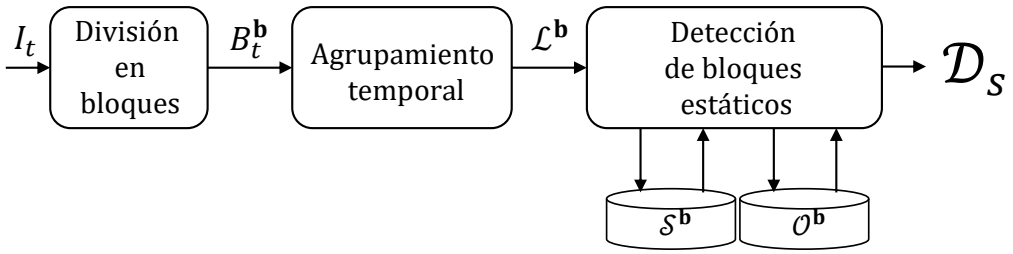



Figura 2.1: Diagrama de bloques de la aproximación planteada.

estabilidad espacio-temporales en instantes regulares de muestreo, y estos se asocian a objetos estáticos. Estos cambios se utilizan para identificar objetos estáticos. En primer lugar, una etapa de división por bloques divide para cada instante $t$ cada *frame* $I_t$ en bloques $B_t^{\mathbf{b}}$ de tamaño $N \times N$, donde $\mathbf{b}$ representa la posición del bloque. A continuación, una etapa de Agrupamiento Temporal (ver Sección 2.2.2) modela cada posición $\mathbf{b}$ a lo largo del tiempo, generando *clusters* que actualizan las particiones de datos generadas en cada posición, denominadas $\mathcal{L}^{\mathbf{b}}$. Esta etapa se encarga de la adaptación temporal a las variaciones de la escena, mediante la asignación de cada nuevo bloque $B_t^{\mathbf{b}}$ a uno de los *clusters* de la partición $\mathcal{L}^{\mathbf{b}}$, o bien añadiendo uno nuevo. Solo se analizan los bloques estáticos $B_t^{\mathbf{b}}$ (sin movimiento respecto a $B_{t-1}^{\mathbf{b}}$), dado que se desea detectar objetos estáticos.

Este agrupamiento proporciona robustez frente a los cambios de iluminación, mediante la utilización de *ratios* a nivel de píxel, capaces de distinguir como iguales dos bloques a pesar de que existan cambios en sus valores de luminancia. Por último, la etapa de Detección de bloques estáticos (ver sección 2.2.3) genera una imagen $\mathcal{D}_s$ que contiene los objetos estáticos, donde $s$ define los instantes de muestreo cada $k$ frames. La información asociada al último *cluster* estable $\mathcal{S}^{\mathbf{b}}$, así como a los antiguos *clusters* estables $\mathcal{O}^{\mathbf{b}}$ y el tiempo de alarma $T$, se emplean para detectar, respectivamente, los cambios de estabilidad espacio-temporales, descartar aquellos causados por bloques ya visualizados (escena vacía o detecciones anteriores) y detectar objetos estáticos que exceden el tiempo de alarma definido. Esta última etapa mejora lo visto en el estado del arte, reduciendo los falsos positivos debidos a la intermitencia del movimiento de ciertos objetos y permitiendo detectar estaticidad incluso para aquellos objetos que no han permanecido visibles durante todo el tiempo $T$. En la Figura 2.3 se representa un ejemplo del análisis de la escena realizado. En la Figura 2.2 pueden observarse los resultados del sistema base en diversos entornos, con los bloques detectados como estáticos agrupados en *blobs* y marcados con rectángulos rojos para poder distinguirlos.

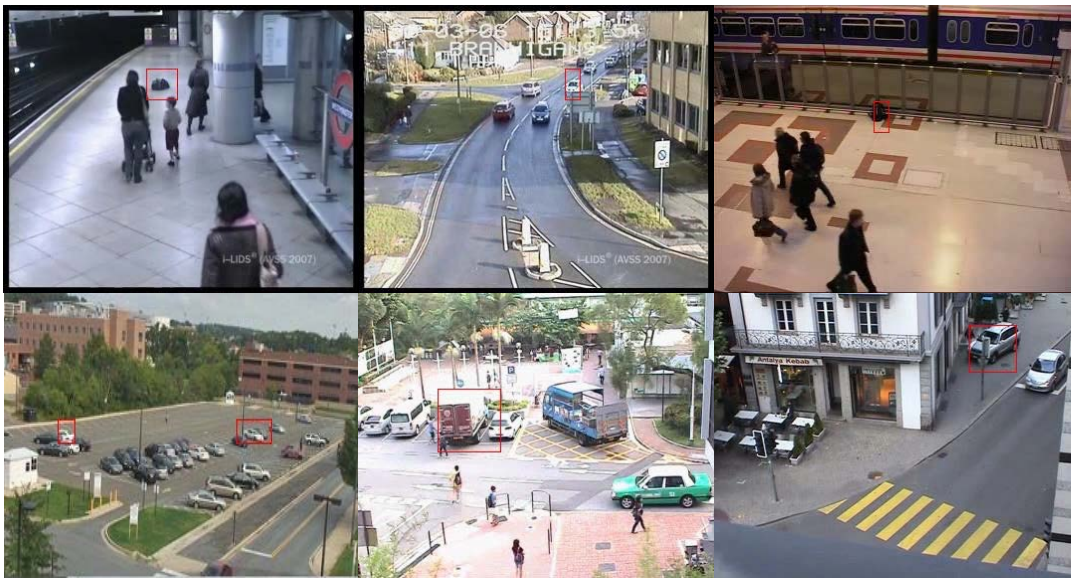


Figura 2.2: Ejemplos de funcionamiento de algoritmos de detección de objetos abandonados en distintos entornos.

### 2.2.2. Agrupamiento temporal

Una vez que cada *frame* $I_t$ se ha divido en bloques no solapados $B_t^{\mathbf{b}}$, la etapa de Agrupamiento temporal modela la evolución temporal de la escena agrupando bloques similares a lo largo del tiempo en *clusters*. Por lo tanto, las particiones de *clusters* $\mathcal{L}^{\mathbf{b}} = \{C_q^{\mathbf{b}}\}_{q=1:|\mathcal{L}^{\mathbf{b}}|}$ se crean para cada posición de bloque $\mathbf{b}$, donde $C_q^{\mathbf{b}}$ es el bloque representante de cada *cluster* y $|\cdot|$representa el número de elementos. Con el fin de obtener una notación más sencilla, se va a omitir el indice $\mathbf{b}$ dado que las operaciones de agrupamiento se aplican en una misma posición.

Teniendo en cuenta que el objetivo es detectar estaticidad, los bloques que contienen objetos en movimiento no son de interés. Por lo tanto, en primer lugar se comparan los bloques entre instantes temporales consecutivos, para descartar aquellos que contienen movimiento, es decir aquellos en los que $B_t$ es distinto a $B_{t-1}$ según la métrica de comparación. Para cada $B_t$ sin movimiento, esta etapa determina con qué *cluster* perteneciente a la partición $\mathcal{L}$ coincide. Cada *cluster* modela un patrón espacio-temporal de la escena y está definido por el primer instante de visualización $f_q$, el último instante de visualización $l_q$, la repetibilidad $w_q$ y el representante del *cluster*, $C_q$. Para actualizar $\mathcal{L}$, si no existe coincidencia con ninguno de los *clusters* existentes, se creará un nuevo *cluster* con un representante $C_{q\prime} = B_t$, donde $q\prime = |\mathcal{L}| + 1$. Por el contrario, si se encuentra una coincidencia con un *cluster* existente, el representante

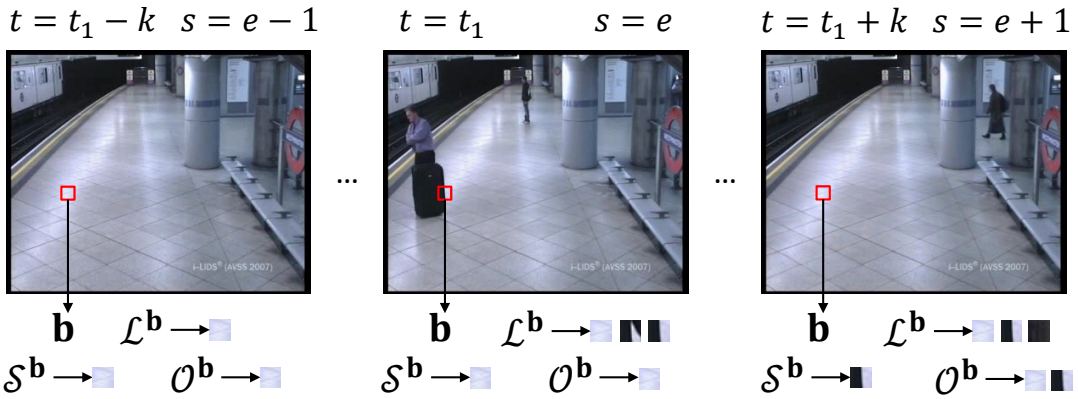



Figura 2.3: Ejemplo del análisis temporal para una posición $\mathbf{b}$ donde cambia la estabilidad desde una escena vacía a una con una maleta. $\mathcal{L}^{\mathbf{b}}$contiene los *clusters* de la posición $\mathbf{b}$, mientras que $\mathcal{S}^{\mathbf{b}}$ y $\mathcal{O}^{\mathbf{b}}$ contienen, respectivamente, la información del último *cluster* estable y los antiguos *clusters* visualizados. Adicionalmente, se muestra relación entre el índice temporal $t$ y el instante de muestreo $s$.

del mismo se actualiza mediante un media móvil acumulativa:

$$C_q = \frac{C_q^{match} \cdot w_q + B_t}{w_q + 1}, \tag{2.1}$$

donde $C_q^{match}$ es el *cluster* que coincide con $B_t$ y $w_q$ es la repetibilidad que cuenta el número de coincidencias $w_q = f\left(C_q, B_{t-T:t}\right)$, disminuyendo su valor $w_q$ en cada cada instante de muestreo $s$ para reducir la contribución de antiguos instantes de visualización. Por ejemplo, dado un instante de muestreo $t$, dicha disminución elimina de la repetibilidad la contribución de antiguas visualizaciones originadas antes del instante $t - T$, es decir, anteriores al periodo correspondiente al tiempo de alarma $T$ desde el instante actual $t$. Este decremento permite una rápida adaptación a las variaciones de la escena, necesaria para operar a largo plazo. Ocasionalmente, $B_t$ puede coincidir con diferentes miembros de $\mathcal{L}$, casos en los que $C_q^{match}$ se selecciona como el $C_q$ tiene la repetibilidad más alta $w_q$. Además, en cada instante de muestreo se reduce $\mathcal{L}$, mediante la eliminación de los miembros con menor repetibilidad, dejando solo los z *clusters* con mayor $w_q$. Por último, cabe destacar que la métrica de comparación entre bloques está basada en la variación estadística de los *ratios* RGB a nivel de píxel de ambos y está detallada en [12].

### 2.2.3. Detección de bloques estáticos

Esta etapa analiza la estabilidad de los *clusters* en instantes de muestreo regulares (cada $k$ *frames*) para identificar estaticidad. Para ello, se guarda la información

asociada a los *clusters* estables antiguos $\mathcal{O}^{\mathbf{b}}$ y el último *cluster* estable $\mathcal{S}^{\mathbf{b}}$ para cada posición de bloque. El primero contiene los *clusters* que han generado detecciones estáticas, i.e $\mathcal{O}^{\mathbf{b}} = \{O_h^{\mathbf{b}}\}_{h=1:|\mathcal{O}^{\mathbf{b}}|}$, mientras que el segundo contiene el último *cluster* estable que, o bien indujo estaticidad, o bien constituye una antigua visualización.

En cada instante de muestreo $s$, se llevan a cabo una serie de operaciones (resumidas en la Figura 2.4 ) para determinar si están abandonados. Primero, el *cluster* más estable de $\mathcal{L}^{\mathbf{b}}$, $C_s^{\mathbf{b}}$, se obtiene como aquel con una mayor repetibilidad $w_q^{\mathbf{b}}$. De esta manera, la existencia de un cambio de estabilidad espacio-temporal se verifica comparando el primer instante de visualización de $C_s^{\mathbf{b}}$ y $\mathcal{S}^{\mathbf{b}}$, es decir $f_s^{\mathbf{b}}$ y $f_*^{\mathbf{b}}$, para detectar si existe un cambio de estabilidad ($\mathbb{S}_s^{\mathbf{b}} = 1$ if $f_s^{\mathbf{b}} \neq f_*^{\mathbf{b}}$) o no ($\mathbb{S}_s^{\mathbf{b}} = 0$ de otra forma). Cabe destacar que el primer instante de visualización es suficiente para verificar la coincidencia entre *clusters*, ya que es una característica única de cada *cluster* en cada posición $\mathbf{b}$. A continuación, se consulta el *buffer* $\mathcal{O}^{\mathbf{b}}$ para determinar si $C_s^{\mathbf{b}}$ es un nuevo *cluster* estable ($\mathbb{N}_s^{\mathbf{b}} = 1$ if $C_s^{\mathbf{b}} \notin \mathcal{O}^{\mathbf{b}}$) o no ($\mathbb{N}_s^{\mathbf{b}} = 0$). Para poder clasificar $C_s^{\mathbf{b}}$ como igual o distinto de $O_h^{\mathbf{b}}$, se utiliza la métrica empleada en la etapa de Agrupamiento temporal. Además, en caso de ser un *cluster* antiguo ($\mathbb{N}_s^{\mathbf{b}} = 0$), el último *cluster* estable $\mathcal{S}^{\mathbf{b}}$se actualiza con $C_s^{\mathbf{b}}$ y $f_s^{\mathbf{b}}$. Cuando $\mathbb{N}_s^{\mathbf{b}} = 1$, se realiza una detección de objeto abandonado:

$$\mathbb{D}_s^{\mathbf{b}} = \begin{cases} 1 & if \quad \left(\mathbb{S}_s^{\mathbf{b}} = 1\right) \wedge \left(\mathbb{N}_s^{\mathbf{b}} = 1\right) \wedge \\ & (lifetime \geq T) \\ 0 & otherwise \end{cases}, \tag{2.2}$$

donde $lifetime = l_s^{\mathbf{b}} - f_s^{\mathbf{b}}$ es la cantidad de *frames* desde el primer al último instante de visualización de $C_s^{\mathbf{b}}$ y $\mathbb{D}_s^{\mathbf{b}} = 1$ denota que $\mathbf{b}$ contiene un nuevo *cluster* estable y por tanto un candidato a objeto abandonado. En ese caso, $C_s^{\mathbf{b}}$, $f_s^{\mathbf{b}}$, $l_s^{\mathbf{b}}$ y $w_s^{\mathbf{b}}$ se asocian al último *cluster* estable $\mathcal{S}^{\mathbf{b}}$ , y se guardan en $\mathcal{O}^{\mathbf{b}}$ como antiguas visualizaciones. Finalmente, como los objetos abandonados pueden expandirse por varios bloques simultáneamente, se realiza una combinación de detecciones vecinas empleando componentes conexas. Estas detecciones forman el conjunto de candidatos a objetos abandonados $\mathcal{D}_s$.

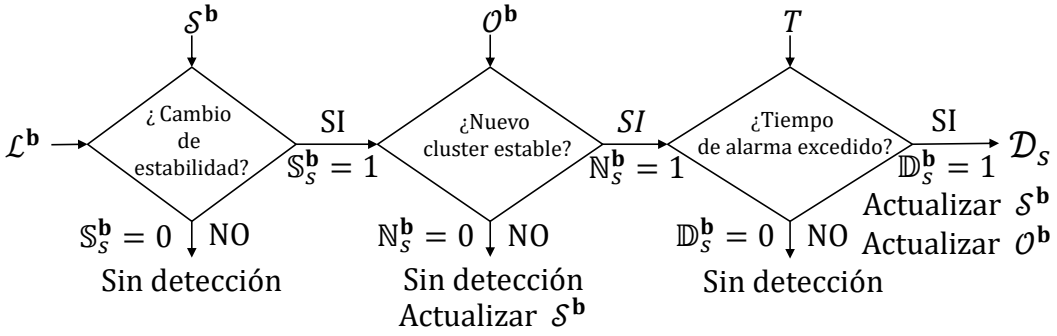



Figura 2.4: Detección de bloques estáticos. Secuencia de operaciones para determinar la estaticidad en un instante de muestreo $s$.

## 2.3. Distributed Video Analysis Framework (DiVA)

En esta sección se describe la plataforma DiVA. Esta plataforma ha sido la diseñada para la integración de los algoritmos de procesado de imagen o vídeo en un sistema común y distribuido, facilitando así la cooperación entre distintos algoritmos y permitiendo el procesado de las fuentes de vídeo en tiempo real. Este sistema ha sido desarrollado por el *Video Processing and Understanding Lab* (VPU) de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid. La estructura general de DiVA puede observarse representada gráficamente en la Figura 2.5.

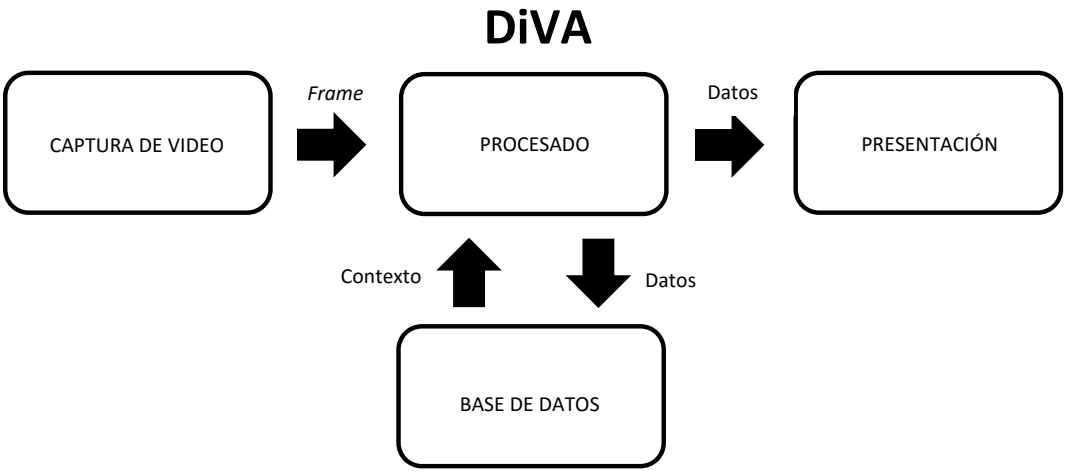



Figura 2.5: Estructura general de funcionamiento e intermcabio de datos entre los distintos módulos estándares de la plataforma DiVA.

Este sistema es escalable (puede añadir módulos de procesado adicionales), eficiente (opera sin apenas incrementar el coste computacional total), generalizable (sus funcionalidades se desarrollan usando herramientas y protocolos genéricos) y con tolerancia a fallos.

Se propone un desarrollo modular en distintos subsistemas. Cada subsistema está relacionado con una función específica dentro de la plataforma. Estas funciones contemplan desde la captura de datos desde distintos dispositivos físicos hasta el almacenaje y presentación de los resultados obtenidos por los algoritmos, todo ello de una forma distribuida.

- **El subsistema de captura de datos** Captura el vídeo de entrada y lo distribuye al sistema *frame* a *frame*. Reserva los recursos necesarios para su funcionamiento.

- **El subsistema de bases de datos** proporciona un contexto de aplicación a los algoritmos de análisis, es decir permite almacenar los distintos parámetros necesarios para la ejecución del algoritmo. Además almacena los resultados del procesado de otros módulos de análisis.

- **El subsistema de procesamiento** realiza el propio procesado de la señal de vídeo y proporciona una interfaz de trabajo para cualquier algoritmo de procesado de vídeo. Está pensando de tal forma que puedan interconectarse en cascada varios módulos de análisis independientes.

- **El subsistema de presentación de datos** presenta en pantalla los datos de los análisis resultantes del subsistema de procesamiento. En ocasiones un único cliente puede realizar las tareas de procesado y presentación.

DiVA utiliza un modelo cliente/servidor en el que se separa la parte servidora de contenido (subsistemas de captura y almacenamiento de datos) de la parte que lo consume (subsistemas de procesamiento y presentación de datos).

# Capítulo 3

# Diseño y desarollo

Este capítulo describe la integración del sistema base en la plataforma de análisis distribuido DiVA, y se se organiza en dos secciones. En la Sección 3.1 se analiza la implementación en C++, mediante la utilización de la librería OpenCV, del sistema original desarrollado en MATLAB. A continuación, en la Sección 3.2 se explica la preparación e integración del algoritmo en DiVA, mediante el encapsulador *DiVA_Algorithm*.

## 3.1. Implementación del sistema base

Para integrar la versión C++ del algoritmo en DiVA se ha creado una clase C++ denominada *SOD_Processor* que implementa la funcionalidad del mismo, formada por diversos módulos cuya funcionalidad describe la propia plataforma. El más importante de estos módulos es el de procesado, que recibe cada uno de los *frames* del vídeo y devuelve los resultados del algoritmo para que otros módulos puedan mostrarlos, guardarlos en disco, o proporcionárselos a otros módulos conectados en cascada. En las siguientes subsecciones se describen los aspectos más relevantes de la implementación C++ del sistema base.

### 3.1.1. Agrupamiento temporal

Esta etapa es la encargada de agrupar las distintas representaciones de la imagen a nivel de bloque que se analizan a lo largo del tiempo. Para modelar la división en bloques de cada imagen y su agrupamiento temporal, se ha empleado un objeto que se ha denominado cubo de bloques. Este cubo es el encargado de almacenar toda la información relativa a la etapa de Agrupamiento temporal.

En concreto, el cubo se ha modelado mediante un vector de dobles punteros a

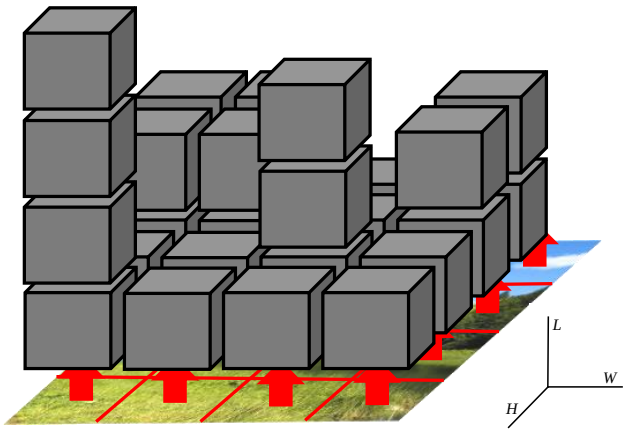



Figura 3.1: Estructura tridimensional del cubo, con dimensiones $H$ y $W$ fijas y cuyo valor depende del tamaño del vídeo, y con la dimensión $L$ variable en cada posición para cada instante temporal determinado.

instancias de una clase anidada en la del algoritmo base, llamada *cluster,* que se define en C++ como $vector < cluster > **cube$. Esta estructura unifica todos las subestructuras de datos que el sistema necesita para trabajar en tiempo real en una sola, facilitando la comprensión del mismo y el manejo de los datos. De esta forma, añadir un nuevo *cluster* en una posición $\mathbf{b}$ consiste simplemente en realizar un *push_ back* del vector de *clusters* presente en dicha posición, añadiendo uno nuevo al final de este, e inicializar sus datos internos.

Esta estructura puede entenderse como una matriz tridimensional con las siguientes características, ilustrada en la Figura 3.1

- Anchura $W$: Número de bloques en que se divide el ancho de la imagen fuente.
- Altura $H$: Número de bloques en que se divide el alto de la imagen fuente.
- Profundidad $L$: Número de *clusters* existentes en un instante t para una posición de la escena $b$. Es decir, la tercera dimensión puede variar para cada posición $b$.

Cada elemento que forma el cubo es un objeto *pcluster*, una instancia de la clase del mismo nombre que modela cada entidad *cluster* descrita en el estado del arte (ver Subsección 2.2.2). Por tanto, cada posición de la escena $\mathbf{b}$, puede verse en el cubo como un vector de dobles punteros a instancias de la clase *cluster,* que se corresponde con la partición $\mathcal{L}_q^{\mathbf{b}}$ descrita en el estado del arte. Un *pcluster* determinado del cubo $\mathcal{L}_q^{\mathbf{b}}$, situado en la posición $\boldsymbol{q}$ del vector de *pclusters* de la posición $\mathbf{b}$ e ilustrado en la Figura 3.2, contiene los siguientes atributos:

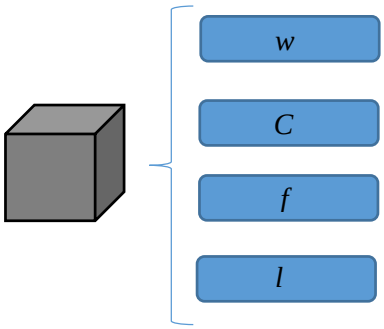



Figura 3.2: Estructura interna de un cluster.

- Repetibilidad $w_q$ : medida de la cantidad de apariciones recientes de un bloque para una posición **b**. En la implemetanción esta es la primera propiedad de la clase *cluster*, modelada mediante una variable tipo *double* ya que puede tomar valores decimales.

- Bloque RGB representante del *cluster* $C_q$ : Bloque RGB que caracteriza al *cluster*. En la implementación se guardan realmente los tres planos de color por separado como tres propiedades separadas de la clase *cluster*, cada una de ellas modelada mediante la estructura de almacenamiento de matrices *Mat* de OpenCV.

- Primer instante de visualización $f_q$: primer instante temporal en que el bloque apareció , es decir, dicho bloque no coincidía con ninguno de los bloques existentes, y se añadió como nuevo. Se modela mediante un entero de precisión simple *int*, dado que al tratarse de un instante temporal siempre va a ser un número entero.

- Último instante de visualización $l_q$: último instante temporal en que un bloque entrante coincidió con el representante del bloque, actualizando el mismo y aumentando su repetibilidad. Al igual que el primer instante de visualización, este también se modela mediante un entero de precisión simple *int*, pues al tratarse de un instante temporal siempre va a ser un número entero.

Existen métodos de la clase *cluster* específicamente diseñados para modificar cualquiera de las propiedades mencionadas, evitando así el acceso directo a las mismas, declaradas en la sección *private* de la clase.

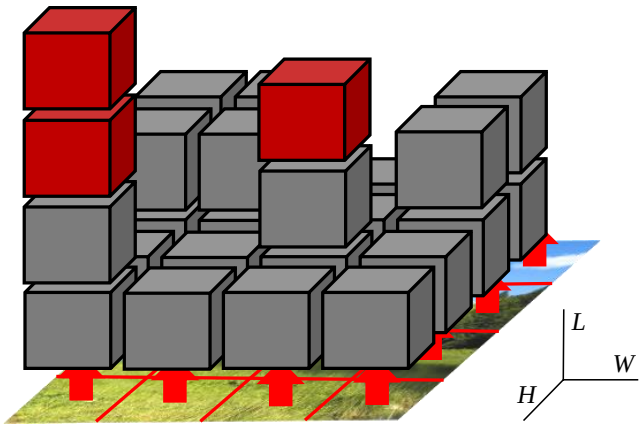



Figura 3.3: Podado del cubo: Se eliminan los *clusters* que exceden un límite prefijado para cada posición **b**, marcados en la figura de color rojo.

### 3.1.2. Detección de bloques estáticos

Mientras que la etapa de Agrupamiento temporal se realiza para cada nuevo *frame* del vídeo, la etapa de detección de bloques estáticos solo se lleva a cabo cada cierto número de *frames* (instantes de muestreo), periodo que define la frecuencia de muestreo. Esta etapa se vale de la información proporcionada por el cubo para determinar si existen objetos estáticos en la escena. En los periodos de muestreo se llevan a cabo dos tareas principales:

1. Podado del cubo: Como se ha mencionado en la sección 2.2.2, según la escena sufre modificaciones se añaden nuevos *clusters* a cada posición **b**. Es necesario limitar el número de *clusters* existentes en cada posición para evitar la ralentización del algoritmo debida al elevado número de comparaciones. Por lo tanto, esta sección, implementada mediante una función del mismo nombre, se encarga de eliminar, para cada posición **b** del cubo, todos los *clusters* a excepción de aquellos tres cuya repetibilidad es mayor. De esta forma se mantiene relativamente estable la cantidad de memoria utilizada por el algoritmo, limitando a la vez el número de comparaciones y, por lo tanto, evitando que el algoritmo se ralentice. Dado que, como se ha mencionado anteriormente, en cada posición **b** existe un vector de instancias de la clase *cluster*, el proceso de podado se implementa de forma muy sencilla en C++ mediante el método *erase* de la clase vector. Esto puede observarse gráficamente en la Figura 3.3, que representa un ejemplo de un instante de podado del cubo.

2. Detección de objetos estáticos: Esta etapa del algoritmo recibe, en cada instante de muestreo, la información contenida en el cubo en ese momento. A partir de esta información, y teniendo en cuenta la situación en instantes previos de muestreo, se decide si existen objetos estáticos en la escena siguiendo el criterio explicado en el estado del arte, es decir, considerando los primeros instantes de visualización de los *clusters* como huella distintiva de los mismos. Si el primer instante de visualización del *cluster* con mayor repetibilidad en una posición **b**, no coincide con el guardado como primer instante de visualización del *cluster* más estable de esa misma posición, denominado *FirstInstantMostStable*, entonces se realiza un filtrado más en cascada, para determinar si se trata o no de un objeto abandonado. En concreto, se consulta una estructura de datos tipo cubo denominada *OldStableClusters,* que almacena antiguos *clusters* estables y que sirve para asegurarse de que el nuevo *cluster* que genere una detección no se haya visualizado antes.

   Siguiendo este mecanismo, se construye *frame* a *frame* una imagen binaria, donde se encuentran activadas (valor 1) aquellas zonas (posiciones **b**) donde existen cambios de estabilidad. A partir de esta imagen, se genera una nueva imagen binaria que asocia espacialmente tanto las detecciones actuales como las detecciones de instantes de muestreo anteriores mediante la extracción de *blobs* de la imagen binaria anterior. Este proceso de asociación espacio-temporal que se realiza mediante la función *FindContours* integrada en OpenCV (ver Figura 3.4) se utiliza para detectar el conjunto de bloques asociados a un mismo objeto.

## 3.2. Integración en DiVA

Para realizar la integración se han llevado a cabo dos niveles:

- Nivel 1: Adaptación del algoritmo a POO (Programación Orientada a Objetos). Para preparar el algoritmo de cara a su integración, se ha encapsulado el algoritmo en una clase de C++ caracterizada por:

  1. Constructor por defecto que inicializa todos los parámetros necesarios para el funcionamiento del algoritmo.
  2. Destructor que libera los recursos de memoria reservados durante la ejecución.
  3. Método *process*, encargado de procesar cada *frame*. La información de contexto requerida de *frames* anteriores, necesaria en algoritmos con memoria

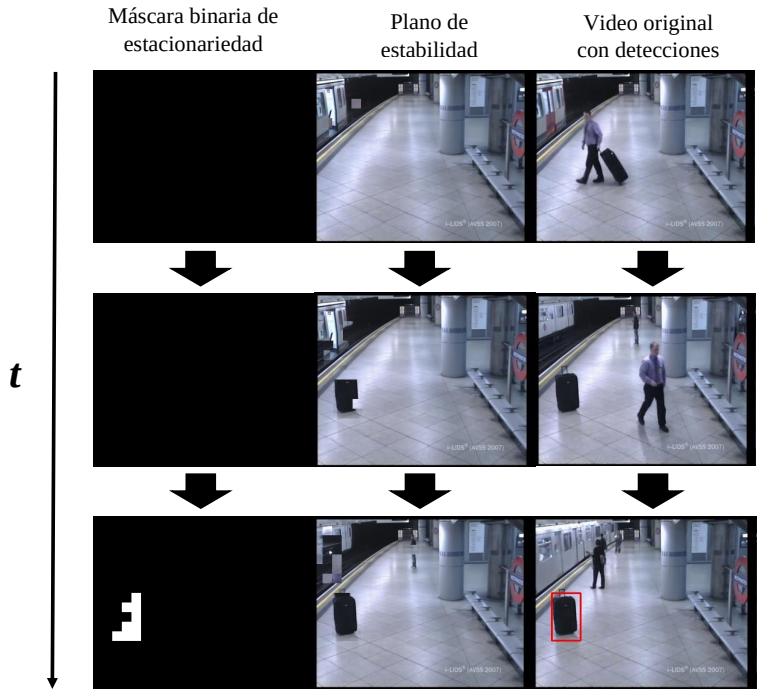



Figura 3.4: Evolución temporal de la máscara binaria de estaticidad ante un objeto estático. Mientras los objetos o personas en la escena no permanecen estáticos durante mucho tiempo, no modifican el plano de estabilidad. Sin embargo, en caso de que se abandone un objeto, acabará generando una detección en forma de *blob* en la máscara binaria.

cuyo resultado no solo depende del frame *actual*, si no tambien de la evolución temporal del vídeo, debe almacenarse en variables internas de la propia clase.

4. La clase contendrá todas las variables y datos necesarios para la ejecución del algoritmo. Las variables serán declaradas como privadas, requiriéndose métodos específicos para su lectura y escritura. En concreto, para el algoritmo sobre el que se trabaja en este proyecto es necesario almacenar tanto la estructura de cubo principal como la relativa a los antiguos *clusters* estables, *OldStableClusters,* o la información sobre el último *cluster* estable, así como numerosas variables de configuración como el tiempo de alarma $T$, el periodo de muestreo o el tamaño de bloque utilizado $N$, entre otras.
5. Método *showresults*, que muestra por pantalla los resultados de procesar cada *frame*. En particular se muestran tanto el plano de estabilidad como el propio vídeo de entrada con las detecciones marcadas.
6. Método *saveresults*, cuya función es guardar un vídeo con el resultado del

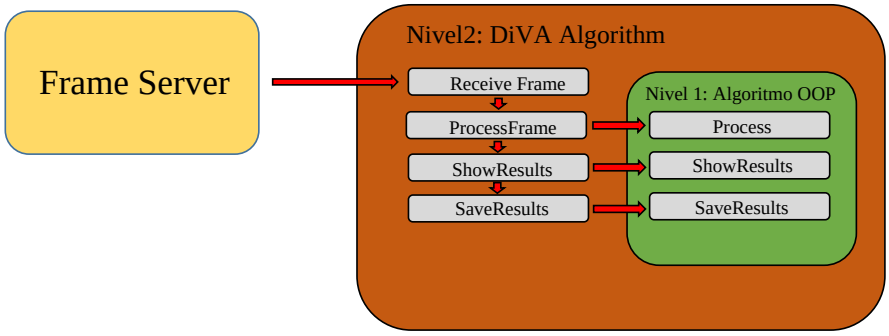



Figura 3.5: Flujo de datos de trabajo de la plataforma DiVA, desde que recibe un frame desde el servidor hasta que el algoritmo elegido lo procesa.

algoritmo para la secuencia analizada, así como otro tipo de archivos de datos que no sean imágenes.

7. Métodos *get_parameter* y *set_parameter*, utilizados para la consulta y modificación de los parámetros del algoritmo, respectivamente.

- Nivel 2: Integración de la clase creada dentro de la clase *DiVA_Algorithm*, encapsulándola mediante la creación de métodos externos que llaman a su vez a los métodos originales de la clase del algoritmo. Este proceso requiere la interconexión de los métodos genéricos de la clase de DiVA con los métodos del algoritmo original. Para realizar esta conversión es necesario convertir los *frames* de vídeo proporcionados por DiVA, que utilizan el antiguo formato de OpenCV para imágenes *IplImage*, al formato *Mat* empleado actualmente. Además, dado que las plantillas de integración proporcionadas en el VPULab fueron originalmente desarrolladas utilizando el antiguo *toolset v100 Windows SDK*, ha sido necesaria la adaptación del código desarrollado originalmente, utilizando las herramientas disponibles actualmente, es decir *toolset v140 Windows SDK*. Esta adaptación ha consistido principalmente en la programación manual de algunas funciones de cálculo científico como *log2f()* o *floor()*, no presentes en el antiguo *toolset*, así como la inclusión de librerias externas para poder disponer de funcionalidades como la estructura *<tuple>* de C++, que permite a una función devolver varios objetos encapsulados en uno solo, y que tampoco estuvo disponible hasta las versiones más recientes del *kit* de desarrollo de *Windows*. El resultado de este proceso de integración y encapsulación puede observarse gráficamente en la Figura 3.5.

# Capítulo 4

# Mejoras del sistema base

Este capítulo se organiza en tres secciones que explican las mejoras introducidas sobre el sistema base para mejorar su rendimiento frente a dos problemáticas comunes para la detección de objetos abandonados: los cambios bruscos de iluminación y las personas estáticas. En primer lugar, se presenta la motivación de introducir estas mejoras, y se ofrece una visión general de las mismas (Sección 4.1). A continuación, se presentan de forma individual los cambios bruscos de iluminación (Sección 4.2) y la detección de personas estáticas (Sección 4.3).

## 4.1. Introducción

El sistema base se diseñó originalmente para operar como una herramienta de detección de objetos estático. Es decir, el algoritmo detecta estaticidad distinta al fondo de la escena independientemente del origen de la misma, ya sea por un objeto, una persona estática, o incluso que un cambio brusco de iluminación pueda provocarlo erróneamente, pues al modificar la luminancia de la escena, es equivalente a ojos del sistema a un cambio en todo el plano . En este capítulo se describen dos mejoras, implementadas sobre el sistema base con el objetivo de refinar las detecciones realizadas.

Teniendo en cuenta que tanto los cambios bruscos de iluminación como la detecciones de personas estáticas son problemáticas que están a la orden del día en los sistemas de video-seguridad, se han abordado de forma que el algoritmo resultante pueda considerarse como de detección de objetos abandonados con mejores capacidades para operar a largo plazo que el original. Así, al sistema base se le han añadido dos módulos adicionales en cascada, que refinan las detecciones originales filtrando aquellas debidas a uno de los dos factores mencionados, para asegurarse de que las

las detecciones finales son realmente generadas por objetos abandonados. Esta modificación del sistema base puede observarse en la Figura 4.1, donde al sistema base se le han añadido dos nuevas etapas: Detección de cambios bruscos de iluminación y Detección de personas.

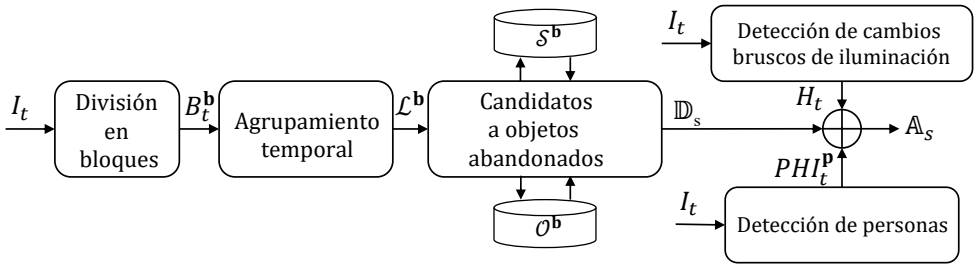



Figura 4.1: Diagrama de bloques de el sistema propuesto.

## 4.2. Cambios bruscos de iluminación

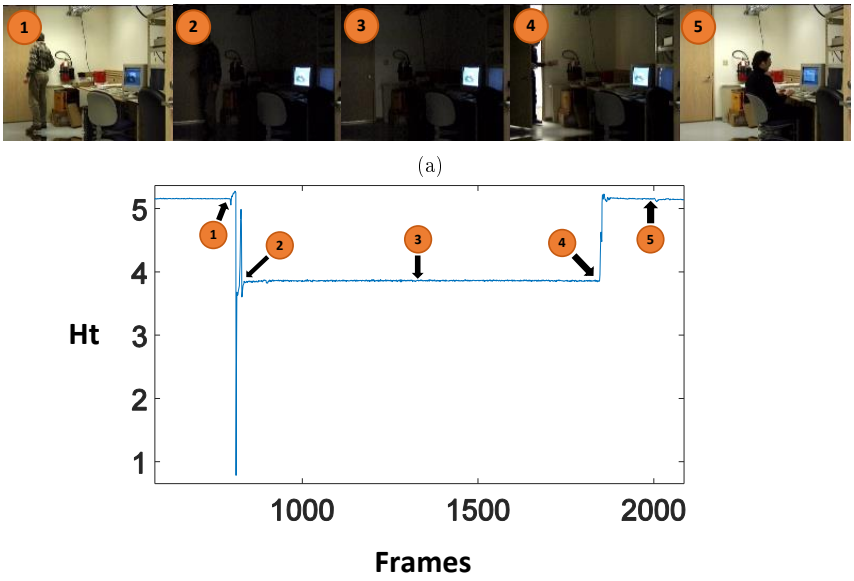
### 4.2.1. Diseño

Un cambio brusco de iluminación se produce cuando una fuente de luz aparece o desaparece repentinamente de la escena, modificándose notablemente la luminancia de la misma a lo largo de varios *frames* del vídeo. Este cambio en la luminancia se comporta a todos los efectos como un cambio de toda la escena, pues la métrica de comparación de bloques no encontrará similitud suficiente entre los antiguos bloques y los nuevos, de forma que para el sistema, toda la escena podría considerarse como un objeto de gran tamaño que ocupa todo el plano, y que acabará generando detecciones una vez excedido el tiempo de alarma.

El cálculo de la entropía de la escena se utiliza para reconocer y evitar las falsas detecciones debidas a cambios de iluminación, de forma similar a la ilustrada en [5]. Según la teoría de la entropía, las imágenes oscuras (claras) presentan valores bajos (altos) de entropía, debido a los valores bajos (altos) de luminancia de sus píxeles. Por lo tanto, el calculo temporal de la entropía es una medida adecuada para detectar cambios bruscos de iluminación, definida como:

$$H_t = -\sum_{l=l_{min}}^{l_{max}} pdf\,(l) \cdot \log\left(pdf\,(l)\right), \tag{4.1}$$

donde $l_{min}$ ($l_{max}$) y $pdf\,(l)$ son, respectivamente, el mínimo (máximo) nivel de luminancia y la función densidad de probabilidad de cada nivel de luminancia $l$ en el *frame* $I_t$. Debemos señalar que $pdf\,(l)$ se calcula como el histograma normalizado de




(a)

(b)

Figura 4.2: Ejemplo de dos cambios bruscos de iluminación. Se apaga la luz durante un periodo de tiempo y como consecuencia el valor de $H_t$ baja pero se mantiene estable. En la imagen (a) pueden observarse cinco instantes temporales de interés, en los que suceden cambios de iluminación, mientras que en (b) se muestra el valor de entropía asociado a dichos cambios.

la luminancia de la imagen. La variación temporal de los valores de entropía de cada *frame* se utiliza para detectar cambios de iluminación:

$$\mathbb{I}_t = \begin{cases} 1 & if \quad |H_t - H_{t-1}| > \alpha \\ 0 & otherwise \end{cases} , \tag{4.2}$$

donde $\alpha$ es un umbral fijado en [5] a 0.05 y en nuestro sistema a 0.15, como se explicará en la Subsección 4.2.2. Por lo tanto, en caso de existir un cambio de iluminación $\mathbb{I}_t = 1$, mientras que en caso contrario $\mathbb{I}_t = 0$. La Figura 4.2 (a) muestra un cambio brusco de iluminación, mientras que la Figura 4.2 (b) representa la entropía asociada a la secuencia de *frames* cercana al cambio, observándose una alta variación de la misma en el momento del cambio de iluminación (justo antes del *frame* 1000).

### 4.2.2. Implementación

El cálculo de la entropía de la luminancia es una operación sencilla a nivel de *frame*, no obstante hay otros detalles sobre esta nueva etapa que es importante detallar. En la práctica, se ha fijado el umbral $\alpha$ de detección de cambios de iluminación a 0.15 en

lugar de el valor original de 0.05 utilizado en [5]. Esta decisión consigue evitar las falsas detecciones de cambios de iluminación debidas al carácter ruidoso de algunos vídeos, así como las debidas al movimiento muy rápido de objetos en la escena, que en caso de disponer de una baja tasa de imágenes por segundo también podrían generar un cambio suficientemente significativo entre dos *frames* consecutivos como para que se considerara un cambio de iluminación. Además, dado que los cambios de iluminación suelen extenderse a lo largo de varios *frames* del video, la variable auxiliar tipo *flag* denominada $\mathbb{I}$ se fija a 1 automáticamente para los siguientes $k$ *frames* después de un cambio brusco de iluminación, siendo $k$ igual al valor del periodo de muestreo descrito en el sistema base. Esta variable $\mathbb{I}$, se almacena dentro de los *clusters* añadidos en la etapa de Agrupamiento temporal, para cada *frame* procesado, a través de una nueva variable interna de la clase *cluster*. De esta forma, cuando un *cluster* determinado vaya a generar un cambio, puede añadirse una etapa de filtrado adicional para comprobar si dicho *cluster* se originó en un instante temporal dentro del intervalo de influencia de un cambio de iluminación, comprobando su valor de $\mathbb{I}$.

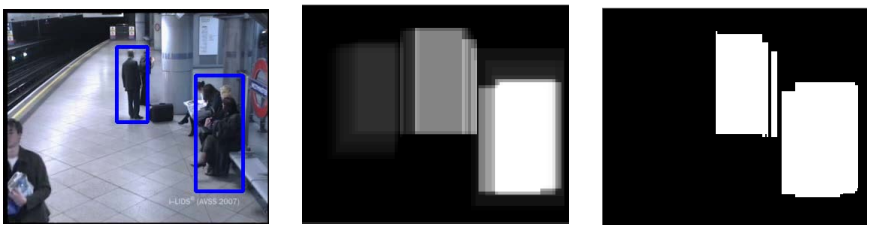
## 4.3. Detección de personas

### 4.3.1. Diseño

Muchos de los candidatos del sistema base a ser objetos abandonados proceden de personas que permanecen estáticas. Para poder filtrar dichas detecciones se utiliza una máscara temporal acumulada (*History Image*) [13] basada en un algoritmo de detección de personas. Este algoritmo genera detecciones rectangulares a partir de las cuales se genera una imagen binaria PM (*Pedestrian Map*) , donde las zonas con peatones se marcan como 1 y las vacías como 0. Las detecciones rectangulares se expanden en todas direcciones con un factor de 0.5, aumentando el área de supresión de detecciones alrededor de las personas detectadas. De esta forma se evitan las detecciones de posibles objetos cercanos a personas que no son de interés al no considerarse abandonados, así como los posibles problemas derivados de las sombras de las personas, que al modificar zonas cercanas a ellas pueden acabar generando falsas detecciones.

La máscara binaria se acumula a lo largo del tiempo, disminuyendo (aumentando) gradualmente su valor en las zonas vacías (con personas), para obtener finalmente la máscara temporal acumulada (*Pedestrian History Image*) $PHI_t^{\mathbf{P}}$:

$$PHI_t^{\mathbf{P}} = PHI_{t-1}^{\mathbf{P}} + \left[w_{pos} \cdot PM_t^{\mathbf{P}}\right] - \left[w_{neg} \cdot \left(\sim PM_t^{\mathbf{P}}\right)\right], \tag{4.3}$$



(a) (b) (c)

Figura 4.3: Ejemplo de cálculo de peatones estáticos. (a) Salida del algoritmo descrito en [6], (b) $PHI_t^{\mathbf{P}}$ y (c) $\mathbb{P}_t^{\mathbf{P}}$.

donde $w_{pos}$ y $w_{neg}$ son pesos usados para controlar la contribución de $PM_t^{\mathbf{P}}$ a $PHI_t^{\mathbf{P}}$. Estos dos pesos deben ser fijados a 1 por defecto (o al valor actual de $PHI_{t-1}^{\mathbf{P}}$) para incrementar (reiniciar) la máscara temporal acumulada. Así, $PHI_t^{\mathbf{P}}$ representa el número de *frames* consecutivos que una detección se ha mantenido activa en $PM_t^{\mathbf{P}}$, mientras que, sin embargo, y dado las detecciones en $PM_t^{\mathbf{P}}$pueden aparecer y desaparecer intermitentemente, se fija $w_{neg}$ a 15 de forma similar a como se hacía en [13] para el calculo de mapas de frente de escena.

Cabe destacar que $PHI_t^{\mathbf{P}}$ está normalizado entre en el rango [0,1], por lo que 1 denota que $PM_t^{\mathbf{P}}$ ha alcanzado el tiempo de alarma. En consecuencia, umbralizando $PHI_t^{\mathbf{P}}$ se pueden detectar personas estacionarias a nivel de píxel, $\mathbb{P}_t^{\mathbf{P}}$, como:

$$\mathbb{P}_t^{\mathbf{P}} = \begin{cases} 1\, if & PHI_t^{\mathbf{P}} \geq \eta \\ 0 & otherwise \end{cases}, \tag{4.4}$$

donde $\eta = 0.5$ es el umbral de estaticidad. La Figura 4.3 representa un ejemplo de detecciones de personas, mostrando además la máscara temporal acumulada ($PHI_t$) y su umbralización. $\mathbb{P}_t^{\mathbf{P}}$ representa píxeles asociados a personas estáticas, permitiendo así refinar la detección de objetos abandonados evitando que aquellos bloques superpuestos total o parcialmente con cualquier zona donde $\mathbb{P}_t^{\mathbf{P}} = 1$ puedan generar detecciones de objetos abandonados.

El proceso completo de detección de objetos abandonados puede observarse gráficamente en la Figura 4.4.

### 4.3.2. Implementación

La detección de personas es una tarea complicada que habitualmente conlleva un alto coste computacional. Dado que uno de los objetivos del proyecto es desarrollar un algoritmo que opere en tiempo real, se ha elegido el sistema de detección de personas descrito en [6]. Este algoritmo destaca por generar detecciones de personas con una

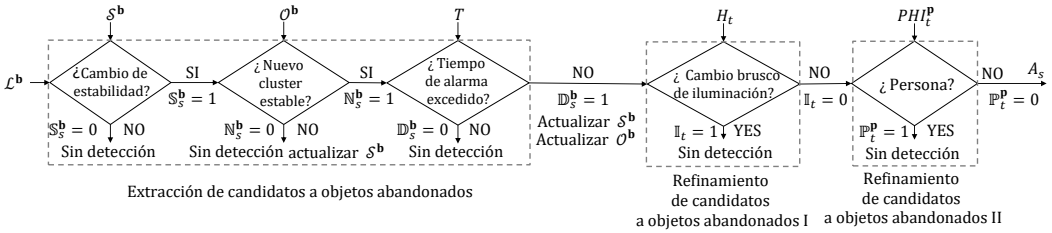



Figura 4.4: Detección de objetos abandonados. Secuencia de operaciones para determinar la existencia de objetos abandonados en un instante de muestreo $s$ para el sistema desarrollado.

buena precisión y conseguir a la vez una elevada velocidad de funcionamiento, por lo que es ideal para este trabajo de cara a conseguir que el sistema final pueda funcionar en tiempo real.

En cuanto a la supresión de detecciones debidas a personas, su implementación es similar a como se hizo para los cambios de iluminación, mediante la inclusión en cada *cluster* de una variable adicional $\mathbb{P}$ (ver Figura 4.5), que indica si el bloque coincide con alguna zona donde $\mathbb{P}_t^{\mathbf{P}} = 1$. De este modo, y a diferencia del mecanismo utilizado para el tratamiento de los cambios de iluminación, un bloque suprimido por su cercanía a una persona puede generar una detección, cuando ésta se va, permitiendo la detección de objetos abandonados próximos a personas que han permanecido estáticas a su lado durante un tiempo antes de desplazarse.

Finalmente, los candidatos a objetos abandonados del sistema base, almacenados en $\mathcal{D}_s$ y que no son filtrados, conforman la imagen resultante $A_s$. Esta imagen contiene las detecciones de objetos abandonados finales, después del filtrado de detecciones no deseadas debidas a cambios de iluminación, así como las debidas a personas estáticas.

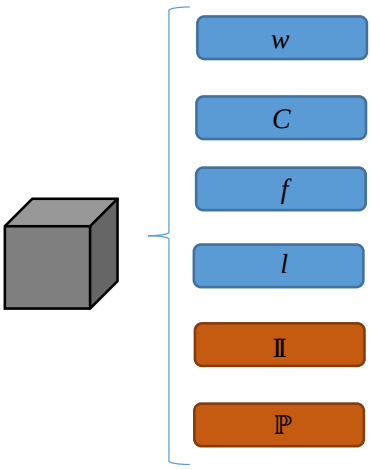



Figura 4.5: Estructura interna de un *cluster* adaptado para tratar los cambios de iluminación y supresión de la detección de personas estáticas.

# Capítulo 5

# Trabajo experimental

Este capítulo presenta las métricas y *datasets* empleados para la evaluación del algoritmo desarrollado (ver Capítulo 3), así como los resultados de dicha evaluación. Los test realizados se han ejecutado en un PC estándar (Intel i7 4770 3.4 GHz, 8 GB RAM).

El capítulo está dividido en dos secciones, una primera donde se introduce el sistema de evaluación y los *datasets* utilizados (Sección 5.1), y una segunda en la que se presentan y analizan los resultados de dicha comparación (Sección 5.2).

## 5.1. Dataset y métricas

En esta etapa se llevan a cabo tres tipos de evaluación, las dos primeras relativas a las mejoras implementadas sobre el algoritmo base en la Sección 4 , para comprobar tanto la robustez contra cambios bruscos de iluminación como la capacidad de filtrar falsas detecciones debidas a personas estáticas en secuencias del estado del arte. Por otro lado, se evalúa el coste computacional de la implementación C++ del algoritmo base comparando las velocidad de ambos algoritmos en *frames* procesados por segundo, utilizando para ello las mismas secuencias empleadas en las dos evaluaciones anteriores.

| Cambios bruscos de iluminación | | | | |
|---|---|---|---|---|
| ABODA | | I2R | LIMU | Wallflower |
| *Video6* | *Video7* | *Lobby* | *LightSwitch* | *LightSwitch* |
| $720 \times 480$ | $720 \times 480$ | $160 \times 128$ | $320 \times 240$ | $160 \times 120$ |

| Personas estáticas | | | | | |
|---|---|---|---|---|---|
| AVSS07 | | | | | PETS06 |
| *AB_E* | *AB_M* | *AB_H* | *PV_E* | *PV_H* | *Cam3* |
| $360x288$ | $360x288$ | $360x288$ | $360x288$ | $360x288$ | $360x288$ |

Tabla 5.1: Resolución y problemática presente en cada uno de los vídeos utilizados.

En primer lugar, para evaluar las dos mejoras del algoritmo propuesto se han seleccionado dos conjuntos de secuencias. Por un lado, para la mejora relacionada con los cambios bruscos de iluminación, se han utilizado 5 secuencias de los *dataset* ABODA [1], I2R [2], LIMU [3] y Wallflower [4], que presentan dicha dificultad. Por otro lado, para la mejora relacionada con la detección de personas, se han seleccionado secuencias del estado del arte que contienen personas de entre los *datasets* AVSS07 [5] y PETS06 [6]. La Tabla 5.1 contiene los detalles de todas las secuencias utilizadas.

Para evaluar el algoritmo propuesto frente al sistema base, se han empleado las métricas de TP (*True Positives*) y AFP (*Accumulated False Positives*) , que representan las detecciones correctas y el área acumulado de falsas detecciones a nivel de píxel, respectivamente.

En cuanto al tiempo de alarma $T$, se ha fijado a 10 segundos para los vídeos de I2R, a 30 segundos para los de ABODA, LIMU y Wallflower, y a 1 minuto para los de AVSS07. Esta variedad de tiempos de alarma en función de la secuencia es necesaria para evitar el enmascaramiento de falsas detecciones como consecuencia de la brevedad de algunas secuencias o del elevado tiempo de detección empleado.

[1] http://imp.iis.sinica.edu.tw/ABODA/index.html
[2] http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html
[3] http://limu.ait.kyushu-u.ac.jp/dataset/en/
[4] http://research.microsoft.com/en-us/um/people/jckrumm/wallflower/testimages.htm
[5] http://www.avss2007.org/
[6] http://www.cvg.reading.ac.uk/PETS2006/data.html

## 5.2. Evaluación comparativa

### 5.2.1. Detección de objetos abandonados

| Algorithm | | Cambios bruscos de iluminación | | | | |
|---|---|---|---|---|---|---|
| | | ABODA | | I2R | LIMU | Wallflower |
| | | *Video6* | *Video7* | *Lobby* | *LightSwitch* | *LightSwitch* |
| [12] | GT/TP/AFP | 1/1/33377 | 1/1/280980 | 0/0/20480 | 0/0/62768 | 0/0/15010 |
| Propuesto | GT/TP/AFP | 1/1/0 | 1/1/90368 | 0/0/0 | 0/0/0 | 0/0/0 |

| Algorithm | | Secuencias con personas estáticas | | | | | |
|---|---|---|---|---|---|---|---|
| | | AVSS07 | | | | | PETS06 |
| | | *AB_E* | *AB_M* | *AB_H* | *PV_E* | *PV_H* | *Cam3* |
| [12] | GT/TP/AFP | 1/1/0 | 1/1/5632 | 1/1/5632 | 1/1/0 | 1/1/10 | 1/1/0 |
| Propuesto | GT/TP/AFP | 1/1/0 | 1/1/0 | 1/1/3584 | 1/1/0 | 1/1/10 | 1/1/0 |

Tabla 5.2: Evaluación comparativa. GT, TP y AFP denotan, respectivamente, número real de detecciones, número de detecciones correctas y el error acumulado en pixels. La aproximación planteada consigue mejores resultados tanto frente a cambios de iluminación como en presencia de personas que ocasionan detecciones estáticas.

El algoritmo propuesto en este TFG se ha comparado con el sistema base [12]. La tabla superior de la Tabla 5.2 muestra los resultados para cambios bruscos de iluminación. El sistema mejorado introduciendo el refinamiento de candidatos a objetos abandonados descrito en 4, detecta todos los objetos abandonados en cuatro secuencias, manteniendo a cero el valor de AFP para todas ellas, mientras que [12] sufre numerosos falsos positivos debido a los cambios de iluminación. El *Video7* de ABODA es el único en el que los resultados del sistema mejorado no son tan buenos, ya que aunque se mejora respecto al sistema base, si aparecen algunos falsos positivos debido al alto nivel de ruido en ausencia de iluminación en la escena. En la Figura 5.1 pueden observarse ejemplos de situaciones donde el algoritmo base cometía errores debido a estos dos tipos de problemática.

Por suparte, la tabla inferior de la Tabla5.2 muestra los resultados para seis se-


Figura 5.1: Ejemplo comparativo del comportamiento del sistema base (derecha), frente al sistema mejorado (izquierda), en situaciones que presentan las problemáticas tratadas.

cuencias con personas. Para los vídeos *AB_E*, *PV_E*, *PV_H* y *Cam3* los resultados obtenidos no varían respecto al sistema base, dado que no se detectan personas estáticas. Sin embargo, para los vídeos *AB_M* y *AB_H* se consigue reducir notablemente el número de falsos positivos presentes en los resultados de [12], causados por personas estáticas, e incluso eliminarlos completamente en el caso de *AB_M,* gracias a la información que proporciona la máscara temporal acumulada (*History Image*) de personas introducida en esta modificación del algoritmo base. La Figura 5.2 representa varios ejemplos de detecciones para las secuencias AVSS07 y PETS06.

### 5.2.2. Coste computacional

La utilización de C++ como lenguaje de implementación del algoritmo de AOD propuesto permite mejoras sustanciales de coste computacional con respecto a el sistema base, que utilizaba MATLAB.

La Tabla 5.3 muestra los *frames* procesados por segundo para cada uno de los vídeos utilizados para comprobar el funcionamiento del sistema base. Además especifica el tamaño de los *frames* de cada vídeo, para entender correctamente el coste computacional del sistema. Como puede observarse, la mejora al utilizar C++ es muy notable, de forma que para una resolución habitual en los sistemas de vídeoseguridad como lo es $360 \times 288$, se procesan según los experimentos el doble de *frames* por segundo que en el sistema base.

La última columna de la tabla muestra el coste computacional del algoritmo mejorado para funcionar como AOD. Se observa un incremento del coste computacional sustancial de en torno a un 25 %, debido principalmente al trabajo en paralelo del al-

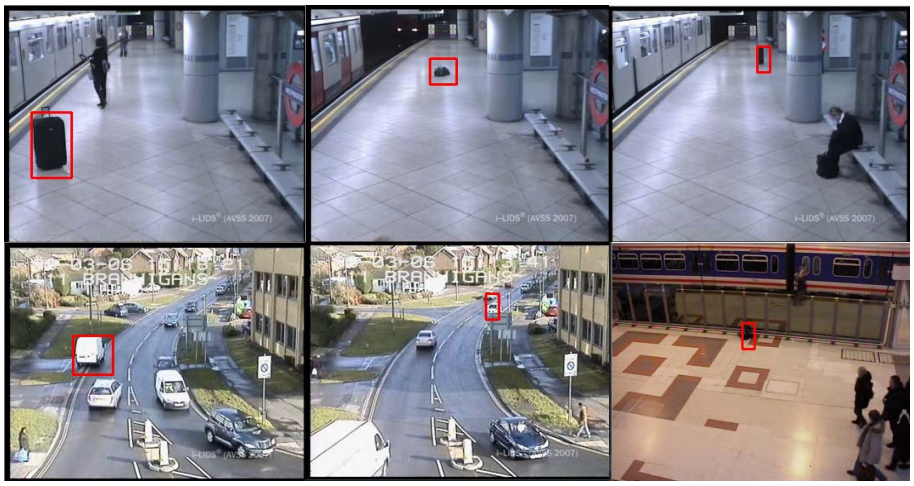


Figura 5.2: Ejemplos de imagenes con resultados $A_s$ para secuencias con peatones.

goritmo de detección de personas ACF introducido en 4.3. Por otro lado, el algoritmo de detección de cambios bruscos de iluminación solo introduce un retardo adicional despreciable, ya que el cálculo de la entropía es un proceso sencillo que solo requiere realizar sumas y multiplicaciones . Además, cabe destacar que a mayor tamaño de imagen, además de el aumento de coste computacional evidente del algoritmo base, también se incrementa el retardo introducido por ACF, al tener que ejecutar la búsqueda de personas en un área mayor, por lo que para resoluciones de imagen muy grandes el rendimiento se deteriora notablemente. En cuanto a la capacidad para operar en tiempo real que se pretendía en este trabajo, si se considera un mínimo de 10 *fps* (*frames per second*), como la velocidad de procesado mínima considerada como tiempo real en vídeo-monitorización, podemos decir que la implementación C++ del algoritmo base mejorada para trabajar como AOD es capaz de ello siempre que la fuente de vídeo no tenga un resolución muy elevada.

| Vídeo | Dataset | Resolución | Matlab system (fps) | C++ system (fps) | C++ AOD system (fps) |
|---|---|---|---|---|---|
| *video 1* | ABODA | $640 \times 480$ | 3.7141 | 8.61183 | 6.0941 |
| *AB_E* | AVSS07 | $360 \times 288$ | 9.9979 | 20.1217 | 14.9253 |
| *AB_M* | AVSS07 | $360 \times 288$ | 9.8840 | 18.9928 | 14.2771 |

Tabla 5.3: Comparativa del coste computacional

# Capítulo 6

# Conclusiones y trabajo futuro

## 6.1. Conclusiones

El objetivo principal de este proyecto era la integración de un algoritmo de detección de objetos abandonados en la plataforma DiVA. Además, existían diversos objetivos secundarios como la optimización de su rendimiento, aprovechando el potencial de C++ como lenguaje de implementación.

En primer lugar, se realizó un estudio detallado del sistema base, de donde se se obtuvieron los conocimiento necesarios sobre su funcionamiento como para realizar la implementación del mismo en C++. Este estudio ayudó además a la hora de optimizar su rendimiento, antes de abordar la integración del algoritmo en DiVA, siguiendo las reglas establecidas por la plataforma.

Tras examinar en profundidad el rendimiento del sistema base, se observó que existía un gran margen de mejora en lo que respecta a la precisión de las detecciones ante ciertas problemáticas típicas en este tipo de sistemas, como situaciones con personas estáticas en la escena o cambios bruscos de iluminación. Estos aspectos se abordaron añadiendo dos módulos de procesado adicionales, añadidos en cascada al sistema base, encargados de refinar las detecciones del mismo para evitar fallos producidos por estos factores.

Por último, para llevar a cabo la evaluación del algoritmo mejorado, se utilizaron vídeos procedentes de diversos *datasets* públicos destinados a la detección de objetos abandonados que además presentaban las dos problemáticas abordadas en este trabajo, personas estáticas y cambios bruscos de iluminación.

## 6.2. Trabajo futuro

La primera de las limitaciones de este trabajo tiene que ver con la velocidad del mismo. A pesar de que se ha conseguido operar en tiempo real para secuencias de video de baja resolución, esto no es posible cuando la resolución vídeo fuente es relativamente elevada, ya que el número de frames procesados por segundo (*fps*) decae. Esta limitación podría abordarse en el futuro de diversas maneras, siendo una de las más efectivas la introducción de una arquitectura multi-hilo que permitiera al algoritmo de detección de personas ACF funcionar en paralelo, así como paralelizar algunas de las operaciones más costosas a nivel de bloque del propio algoritmo principal.

Además, y a pesar de que los módulos de filtrado adicional consiguen aumentar la precisión de las detecciones de objetos abandonados, siguen existiendo falsos positivos, debidos a personas que no son detectadas por el algoritmo ACF al estar sentadas u ocluidas por otra persona u objeto, o bien debidas a cambios bruscos de iluminación no detectados o demasiado largos en el tiempo para que la estrategia de supresión temporal surta efecto. Por tanto, aún existe margen de mejora en lo que se refiere a la precisión de las detecciones, que podría abordarse mediante el perfeccionamiento de los algoritmos de filtrado y supresión de detecciones implementados en este trabajo.

Por último, cabe destacar la conveniencia de crear una interfaz gráfica para el sistema, cuya incorporación aportaría numerosos beneficios, como mayor facilidad de uso, selección del vídeo fuente, visualización de resultados, modificación de los parámetros de configuración, etc. Esta aportación sería de especial utilidad a la hora de permitir que usuarios sin conocimientos de programación y tratamiento de imagen pudieran utilizar el sistema con facilidad.

# Anexo A
# Publicaciones

Parte de este trabajo ha sido incluido en la siguiente publicación:

- Sergio López , Diego Ortego, Juan C. SanMiguel, José M. Martínez: "Abandoned Object Detection under Sudden Illumination Changes", en XXXI Simposium Nacional de la Unión Científica Internacional de Radio, Detección y Seguimiento de Objetos en Secuencias de Vídeo (DSOSV), URSI 2016.

# Bibliografía


[1] A. Albiol, L. Sanchis, A. Albiol, and J.M. Mossi. Detection of parked vehicles using spatiotemporal maps. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1277–1291, Dec 2011.

[2] A. Bayona, J.C. SanMiguel, and J.M. Martinez. Comparative evaluation of stationary foreground object detection algorithms based on background subtraction techniques. In *Proceedings of IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 25–30, Sept 2009.

[3] A. Bayona, J.C. SanMiguel, and J.M. Martinez. Stationary foreground detection using background subtraction and temporal difference in video surveillance. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pages 4657–4660, Sept 2010.

[4] T. Bouwmans. Traditional and recent approaches in background modeling for foreground detection: An overview. *Computer Science Review*, 11-12:31–66, 2014.

[5] F. C. Cheng, S. C. Huang, and S. J. Ruan. Illumination-sensitive background modeling approach for accurate moving object detection. *IEEE Transactions on Broadcasting*, 57(4):794–801, Dec 2011.

[6] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, Aug 2014.

[7] R.H. Evangelio and T. Sikora. Complementary background models for the detection of static and moving objects in crowded environments. In *Proceedings of IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 71–76, Aug 2011.

[8] Q. Fan, P. Gabbur, and S. Pankanti. Relative attributes for large-scale abandoned object detection. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 2736–2743, Dec 2013.

[9] S. Guler, J.A. Silverstein, and I.H. Pushee. Stationary objects in multiple object tracking. In *Proceedings of IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 248–253, Sept 2007.

[10] J. Kim and D. Kim. Accurate static region classification using multiple cues for ARO detection. *IEEE Signal Processing Letters*, 21(8):937–941, Aug 2014.

[11] K. Lin, S. Chen, C. Chen, D. Lin, and Y. Hung. Abandoned object detection via temporal consistency modeling and back-tracing verification for visual surveillance. *IEEE Transactions on Information Forensics and Security*, 2015.

[12] D. Ortego, J. C. SanMiguel, and J. M. Martínez. Long-term stationary object detection based on spatio-temporal change detection. *IEEE Signal Processing Letters*, 22(12):2368–2372, Dec 2015.

[13] D. Ortego and J.C. SanMiguel. Multi-feature stationary foreground detection for crowded video-surveillance. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pages 2403–2407, Oct 2014.

[14] J. Pan, Q. Fan, and S. Pankanti. Robust abandoned object detection using region-level analysis. In *Proceedings of IEEE International Conference on Image Processing (ICIP)*, pages 3597–3600, Sept 2011.

[15] F. Porikli, Y. Ivanov, and T. Haga. Robust abandoned object detection using dual foregrounds. *EURASIP Journal on Advances in Signal Processing*, Article ID 197875, 2008.

[16] G. Szwoch. Extraction of stable foreground image regions for unattended luggage detection. *Multimedia Tools and Applications*, pages 1–26, 2014.

[17] Y. Tian, R. S. Feris, H. Liu, A. Hampapur, and M. T. Sun. Robust detection of abandoned and removed objects in complex surveillance videos. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(5):565–576, Sept 2011.

[18] T. YingLi, R.S. Feris, L. Haowei, A. Hampapur, and S. Ming-Ting. Robust detection of abandoned and removed objects in complex surveillance videos. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 41(5):565–576, Sept 2011.